

第1章

組み込み分野における「マルチプロセッサ」とは

―― 多機能・低消費電力の要求にこたえるための技術的要素と課題

吉田正康

ひと言で「マルチプロセッサ」と言っても，ハードウェアやソフトウェアの構成などによって実現できる機能が異なる．また，マルチプロセッサ構成を採用すると性能の向上や消費電力の低減に効果があるが，その一方で，開発環境などの整備の点で課題が残っている．ここでは，こうしたマルチプロセッサの基本的な構成や利用のメリット，今後の課題について，とくに組み込み機器分野での利用を意識して解説する．（編集部）

近年，ゲーム機やパソコンなど，高い性能が要求される民生機器の分野において，「マルチプロセッサ」や「マルチコア（1チップ上に集積したマルチプロセッサ・システム）」ということばを耳にすることが増えました．携帯電話やカー・ナビゲーション・システムなどでもさらなる高性能化が求められており，マルチプロセッサに対する期待が高まってきています．ここでは，まずマルチプロセッサの概要を解説します．次に，マルチプロセッサを組み込み機器分野で適用する際の課題と展望を説明します．

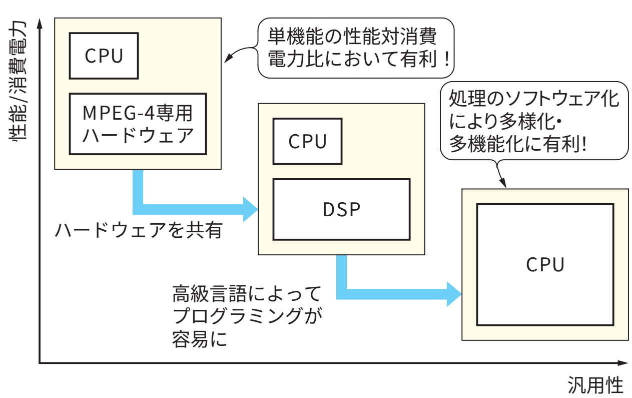


図1　プロセッサのメリット，デメリット

例えば，アプリケーションとしてMPEG-4 CODECを高速CPUですべて処理した場合，DSPを使って処理した場合，MPEG-4専用ハードウェアを用いて処理した場合を考える．一般に，MPEG-4のような信号処理アプリケーションでは，汎用のCPUよりも積和演算器を備えたDSPを使ったほうが同じ処理を少ない消費電力で実行できる．また，MPEG-4専用のハードウェアを作ると，消費電力はさらに少なくなる．これに対して，MPEG-4以外のアプリケーションへの適用を考えると，専用ハードウェアはもっとも汎用性が小さく，CPUがもっとも大きくなる．つまり，汎用性と単体アプリケーションの性能対消費電力比はトレードオフの関係になる．したがって，汎用性を重視するパソコンでは高速CPUが向いているが，機能の限定された組み込み機器では性能が必要な部分を専用ハードウェアやDSPで処理させ，残りを小さなCPUで行うというようなマルチプロセッサ形態が有利になる．

1　組み込み向けのマルチプロセッサとは

一般に白物家電やAV機器などの組み込み機器分野では，あらかじめ決められたことを決められた手順で，かつ最小限のコストで実現することが求められています．このため，これまで信号処理はおもに専用のハード・ワイヤード論理で実現されてきました．しかし，携帯電話やカー・ナビゲーション・システムなどの一部の組み込み機器では，機能の複合化に伴ってシステムが複雑かつ大規模になっており，機能の組み合わせをいかに効率良く実現するかが問題になってきています．また，とくに携帯電話などでは，市場の急激な立ち上がりに対応するため，非常に短い期間での開発が求められています．

これらの問題に対する解として，近年，DSP（ディジタル信号処理プロセッサ）やCPUなど，ソフトウェアのプログラミングによって実現する論理機能を利用する方法が重要視されてきています．本稿では，こうした方法を「プロセッサ」と定義したいと思います．

● プロセッサを用いたシステム設計の一長一短

プロセッサはハード・ワイヤード論理と比べて，以下のようなメリットがあります（図1）．

- さまざまな機能に対してソフトウェアを再利用すること

によって，システム全体のハードウェア規模を低減できる（とくに大規模システムにおいて再利用率が高くなる）

- 回路設計後に発見された不ぐあいをソフトウェアで回避・修正できる
- システム設計後に，機能の追加や個人の要望に合わせたカスタマイズが行える

しかし，その反面，以下のようなデメリットがあります．

- 回路が冗長になり，単一機能においては性能や消費電力の面でかならずしも最適値ではない
- 開発にあたって，ハードウェアとソフトウェアの両面を設計する必要があり，開発の複雑度が増す

すなわち，プロセッサを用いてシステム設計を行う場合は，次の2点に注意することが重要です．

1）性能や消費電力などを事前に検証することで，必要十分なマージンを明確にしておく

2）ハードウェアとソフトウェアの両面を開発，デバッグできるツール群をあらかじめ準備しておく

● 組み込み向けマルチプロセッサはヘテロで非対称な構成

近年，組み込み機器向けプロセッサの分野でも，マルチプロセッサが脚光を浴びています．

そもそも製品としての組み込みシステムは，単一の処理ではなく，さまざまな処理が分散・協調しながら実行されるものがほとんどです．例えば携帯電話端末には，通信機能やゲーム，ユーザ・インターフェースといったアプリケーション機能が搭載されており，これらの機能は複数のプロセッサ（CPUやDSP）で実現されています．つまり，このような機能を一つの装置として実現しているシステムは，必然的にマルチプロセッサ構成をとることになります．

ところで，上記の「マルチプロセッサ」と最近パソコンやゲーム機で使われているコンピュータとしての「マルチプロセッサ」では，少し論点が異なるように感じます．この点について，以下の三つの視点で整理したいと思います（図2）．

1）機能分散と負荷分散

マルチプロセッサを採用したシステムを考えるとき，それぞれのプロセッサが別々の機能を持ち，それらの処理が協調することによってシステムを構築するという考えかたがあります．これを「機能分散」と言います．これに対して，単一の処理を複数のプロセッサで分担することにより，ひとつひとつのプロセッサの処理負荷を減らすことができます．このような考えかたを「負荷分散」と言います．

この二つは，システム設計において意識すべき着眼点といえます．

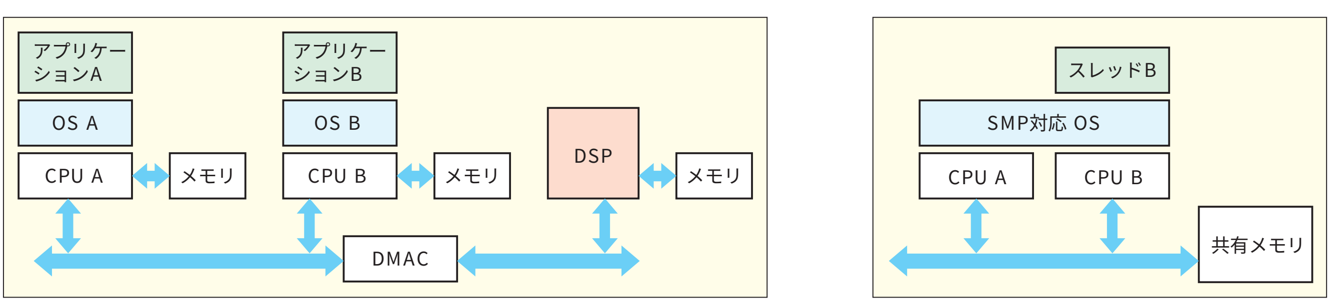
2）ヘテロジニアスとホモジニアス

ヘテロジニアスとホモジニアスということばは，一般にマルチプロセッサのハードウェア構成を説明するときに用いられます．ヘテロジニアスは，プロセッサが「異質な」結合をしているシステムです．例えば，CPUとDSPといったまったく異なるプロセッサを結合しているケースや，CPUどうしであっても異なるアーキテクチャのものを結合しているケースを指します．

一方，ホモジニアスは「等質」な結合をしているシステムを指します．このシステムでは，同一のアーキテクチャや

分類法	機能分散	負荷分散
ハードウェア均質性	ヘテロジニアス（不均質）	ホモジニアス（均質）
ソフトウェア対称性	非対称（AMP）	対称（SMP）
結合距離/レイテンシ	疎結合（遠い）	密結合（近い）
並列化粒度	粗粒度	細粒度

（a）分類



（b）機能分散型（ヘテロジニアス，非対称）　（c）負荷分散型（ホモジニアス，対称）

図2　マルチプロセッサの分類

組み込み分野では，自然発生的に機能分散型構成（ヘテロジニアス，非対称）が発展している．一方，コンピュータの分野では負荷分散型構成（ホモジニアス，対称）が発展している．



構成のプロセッサを複数結合しています．

3）非対称（AMP）と対称（SMP）

非対称か対称かは，ソフトウェアがそれぞれのプロセッサをどう扱うかを説明する視点です．

非対称マルチプロセッサ（AMP）では，それぞれのプロセッサの上でOSに代表される別々のソフトウェア基盤が動作します．したがって，各プロセッサのソフトウェア基盤の上で動く上位ソフトウェアが，それぞれの処理結果を個別に調停しつつ，ほかのプロセッサのソフトウェア基盤と協調動作する必要があります．

対称マルチプロセッサ（SMP）では，各プロセッサ間の処理の調停を単一のソフトウェア基盤が行います．したがって，アプリケーションなどの上位ソフトウェアから見たとき，マルチプロセッサであることを意識しなくてすみます．

先ほど例を挙げた携帯電話における通信プロセッサとアプリケーション・プロセッサなどのようなマルチプロセッサ構成は，もともと別々の機能処理（専用のハード・ワイヤード論理で行っていたもの）をプロセッサ処理に置き換えることで自然発生しました．このような機能分散型のマルチプロセッサでは，必然的にヘテロジニアスな非対称構成をとります．

しかし，コンピュータにおけるマルチプロセッサは，主として高性能と低消費電力の実現の要求に基づき，負荷分散を行うことに主眼を置いています．このため，ホモジニアスで対称な構成を目ざしたアーキテクチャとなっています．

● 複数のプロセッサで性能が向上し，消費電力が下がる

マルチプロセッサによる性能向上の要因としては，以下のものが挙げられます．

- 処理のスループット向上
- レスポンスの向上

処理のスループット向上とは，複数プロセッサの同時動作によるピーク性能の向上を指しています．

レスポンスの向上とは，マルチタスク環境下で動作しているシステムにおいて，タスク応答待機しているプロセッサが増えることにより，ほかのタスクの終了を待つ可能性が低下することです（**図3**）．

また，マルチプロセッサ構成をとると低消費電力化を図ることができます．さて，これはなぜでしょうか．

これまで，プロセッサの高性能化は主として周波数の向上によって実現されていました．プロセッサの設計者はこのために以下のような改善を行ってきました．

- トランジスタの微細化による負荷容量の低減とトグル周波数（トランジスタのスイッチング速度）の向上
- 論理ゲートのファンアウト数の低減とゲート遅延の圧縮
- プロセッサ演算器のパイプライン化によるスループットの改善

この中で，とくに「トランジスタの微細化」と「トグル周

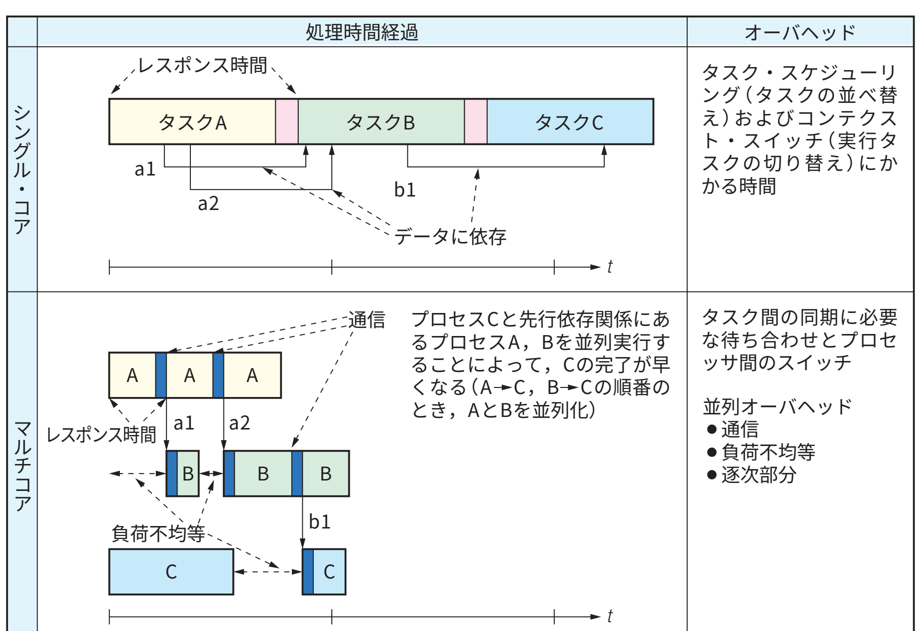


図3
マルチプロセッサによるレスポンスの改善
シングル・コアの場合と比べて，マルチコアでは次のスレッドに対するレスポンス時間を短縮できる．

波数の向上」を両立させるためには，相対的に大電流が流れるようにLSIを製造するか，もしくは相対的に高い電圧で駆動する必要があります．これらの設計は消費電力を大きくする方向に働きます．マルチプロセッサでは，単一プロセッサと同じ性能を実現するために低い動作周波数のプロセッサを複数用いることで，消費電力の増加につながる製造や駆動に関する問題を回避します．

2 マルチプロセッサを支える技術

従来，大規模コンピューティングを中心に採用されていたマルチプロセッサが，組み込み機器でも使えるようになってきた背景には，以下のようなものが挙げられます．

● トランジスタの微細化による機能集積

現在，組み込み機器向けの半導体には，一般に設計ルールが130nmの製造プロセスが用いられています．これが，2005年から2007年にかけて90nmに移行していきます．さらに，その先を見越して65nmプロセスの開発が進んでいます．130nm→90nm→65nmとトランジスタの微細化が進むと，同一面積上に搭載できるトランジスタ数は約2.1倍，それからさらに約1.9倍と増えていきます[注]．

つまり，同等機能で同等の価格を維持するという前提であれば，性能向上に使えるトランジスタ数は飛躍的に向上していくということになります．

● 高機能OSによるマルチタスク処理の普及

とくに，負荷分散型のマルチプロセッサにおいて，プロセッサが実行するタスクやスレッドの処理を割り付けるOSの存在は必須です（下掲のコラム「プロセス，タスク，スレッド」を参照）．

近年，組み込み機器分野においても，OSによる複数のタスク処理（マルチタスク）のスケジューリングは一般的になっています．中でもLinuxやWindowsといった高機能なOSを用いれば動的にタスクを配分することができ，完全な対称マルチプロセッサに対応した処理が期待できます．

なお，機能分散型マルチプロセッサでは，かならずしもOSの機能は必須ではありません．

● ボトルネックを防ぐバス技術の進化

マルチプロセッサでは，プロセッサ間の通信，共通リソースへのアクセスにおいて，複数プロセッサからのバス・アクセス要求が競合する可能性があり，一般に単一プロセッサよりもバスに高い性能が要求されます（キャッシュを持ったプロセッサでは，単一プロセッサよりもキャッシュのリプレイスを低減できるという考えかたもある）．

注：トランジスタの微細化によって，LSIのサイズがかならず小さくなるとは言い切れない．実際には，LSIの出力端子数の関係で，LSIパッケージのサイズが変わらないケースが多々ある．

1

Column プロセス，タスク，スレッド

以下の用語は，並列処理を行うソフトウェア開発で使われますが，混乱されやすいことばです（**図A-1**）．

● プロセス

単体で実行可能なプログラムの単位です．複数のプロセスが存在するときは，それぞれ独立したアドレス空間を持ちます．

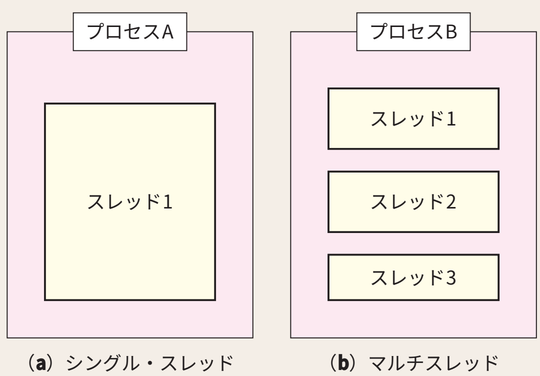
● タスク

OSの処理単位を表します．UNIX系の表現ではプロセスと同義に扱われています．一つのOS上で複数のタスク処理を行えるものを，マルチタスクと呼びます．

● スレッド

プロセスやタスクより細かいプログラムの単位です．スレッド・プログラミング・モデルとしては，一つのプロセスに一つのスレッドを持つ「シングル・スレッド」と，複数のスレッドで構成される「マルチスレッド」があります．

同一プロセス内のスレッドは，アドレス空間，メモリ，ファイル記述子，UID（アクセス制御）を共有します．また，個々のスレッドは，プログラム・カウンタ，スタック，レジスタを保有しています．



図A-1 プロセスとスレッド
一つのプロセスは，一つまたはそれ以上のスレッドで構成される．



バスの性能に対する要求は，ホモジニアスかヘテロジニアスか，対称か非対称かによって異なります．一般に，ホモジニアスな対称マルチプロセッサでは高いバス性能を要求されます．

現在，さまざまな高速バスが提案されていますが，代表的な高性能オンチップ・バスとして，英国ARM社のAXI（Advanced eXtensible Interface）と，オープン規格のOCP（Open Core Protocol）が挙げられます．

●開発を効率化するデバッグ技術の普及

現在のプロセッサの多くは，JTAGをベースとしたデバッグ・ポートをチップ上に有しています．これによって，処理の実行・停止といった制御や処理内容のトレース機能を実現できます．また，これらのデバッグ・ポートを利用したさまざまな製品（デバッガなど）が販売されています．

JTAGを利用すると，複数のデバイスを一つのデバッグ・ポートにシリアルに接続できます．こうした特徴を利用して，マルチプロセッサ対応のデバッガ（マルチプロセッサであっても一つのデバッガに接続できる）をいくつかのベンダが開発しています．

● H.264のデコード処理の消費電流は120mA

上述のハードウェアやソフトウェアの技術の進歩に伴って，マルチプロセッサ構成を1チップに集積した組み込みLSI製品が市場に出始めました．

例えば，**図4**は，筆者ら（NECエレクトロニクス）が開発した携帯電話向けアプリケーション・プロセッサ「MP211」です．本プロセッサには三つのCPU（200MHz動作のARM926）と一つのDSPが集積されており，ヘテロジニアスな非対称マルチプロセッサを構成しています．130nmのCMOSプロセスで製造しています．6,658万個のトランジスタを8.95mm × 8.95mmのチップに集積しており，量産可能なダイ・サイズとなっています．本アプリケーション・プロセッサでは，H.264圧縮画像の復号化処理を200MHzで動作可能なCPUとDSPを用いて実現しています．消費電力は，1.2Vの電源電圧で120mWときわめて低くなっています．

また，ARM社は，完全なホモジニアスで，かつ対称マルチプロセッサのアーキテクチャを実現した「ARM11 MPcore」を発表しています（**図5**）．さらに効率的なマルチプロセッサが組み込み機器の分野で使われようとしています．

3 組み込み機器向けのマルチプロセッサの課題と展望

以上，組み込み機器の分野におけるマルチプロセッサについて簡単に説明しましたが，とくにホモジニアスな対称

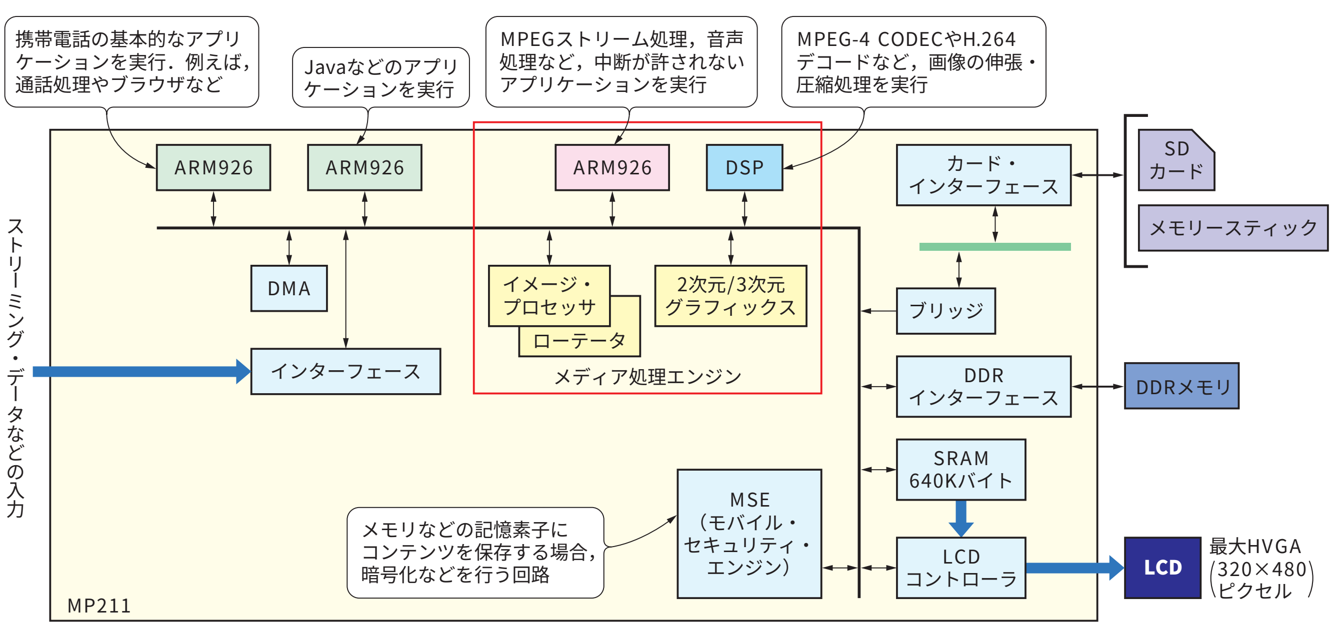


図4　マルチプロセッサの例（その1）

マルチプロセッサを搭載したNECエレクトロニクスの携帯電話向けアプリケーション・プロセッサ「MP211」の内部ブロック図を示す．各プロセッサに最適な処理を割り当てて，高性能・高効率を実現した．

マルチプロセッサ技術はまだ導入初期であり，実際の適用にあたっていくつかの課題があります．以下に，おもな課題とその対策について簡単に説明したいと思います．

● シングル・タスクの性能実現に向けた投機実行

対称マルチプロセッサは，マルチアプリケーション，マルチタスクの処理の加速に適したアーキテクチャですが，組み込み機器においては，かならずしもつねにマルチタスクになるとは限りません．この場合，タスクをより小さな単位であるスレッドに分割して並列動作させることが可能です．しかし，組み込み機器においてはスレッド・プログラミングが適用されていないものがあり，注意が必要です．

なお，本件に関する新しい動きとして，並列実行が可能なスレッドをコンパイラが自動抽出してプロセッサが投機的に実行していく，という考えかたがあります．図6に筆者らの研究内容を簡単に説明します．これにより，マルチプロセッサの負荷分散の効率を最大限に発揮することができます．

● システム性能の最適化のため負荷のバランスを考えて

分割されたタスクやスレッドの負荷のバランスが悪いと，特定のタスクまたはスレッドがボトルネックとなって負荷が適正に分散しません．このため，各タスクやスレッドのバランスを考えたソフトウェアの設計・開発が必要になります．

それぞれのタスクやスレッドの性能は，厳密にはハードウェアに実装したときに初めて決まります．このため，机上でシステム性能を見積もるためのハードウェアのエミュレーション環境や，実ハードウェアの性能を正確に計測するツールが必要になります．このようなツールは，図7の例のようにいくつか出始めていますが，まだまだ普及にはいたっていません．

● 広く普及するためにはデバッグ環境の整備が必須

タスクやスレッドを実行するプロセッサがシステムの稼働中に決定されるような方式を動的負荷分散と言います．このようなシステムでは，特定のプロセッサ動作時にのみ発生する不ぐあいの再現やデバッグが困難になります．

また，一つのプロセッサで不ぐあいが発生すると，ほかのプロセッサの動作によっては，不ぐあいを起こした環境が破壊されてしまい，不ぐあいの再現や解析に支障をきたすことがあります．

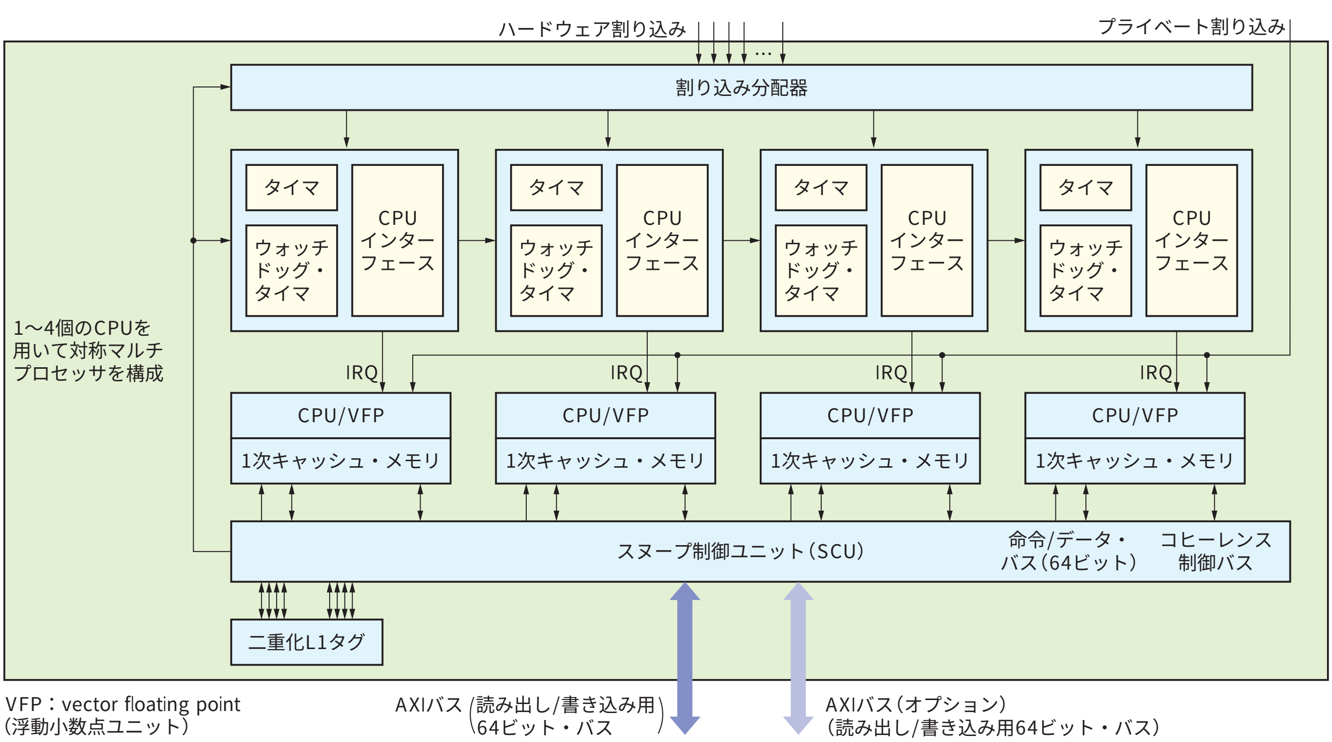
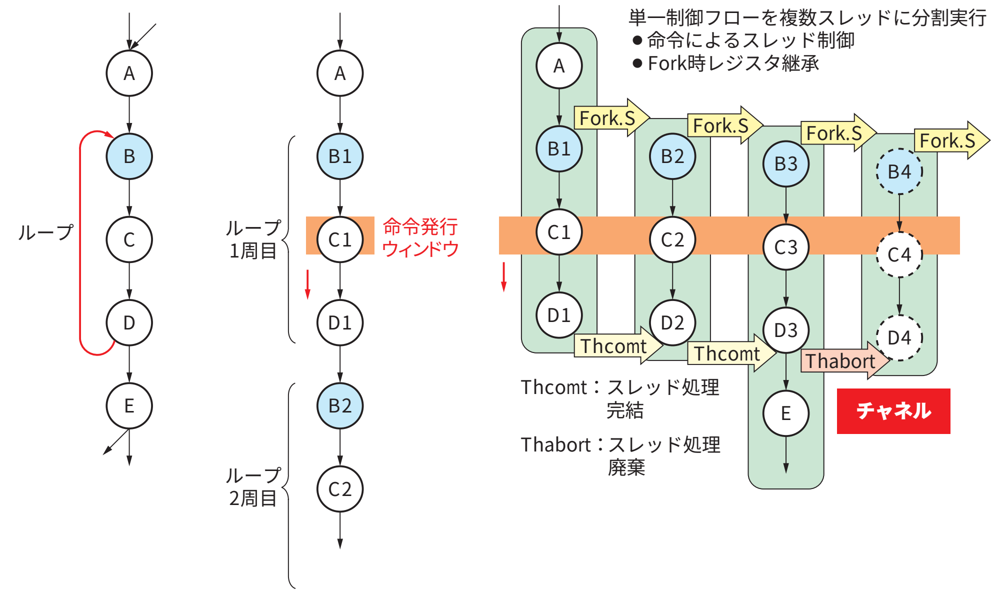


図5　マルチプロセッサの例（その2）

ARM社のマルチプロセッサ・コア「ARM11 MPCore」の内部ブロック図．ホモジニアスで対称マルチプロセッサの処理に対応している．





（a）プログラム
記述構造
（制御フロー・グラフ）

（b）通常のプロセッサ
での実行順序
（単一制御フロー）

（c）投機マルチスレッディング
（制御並列）

図6
自動並列化による投機並列実行
自動並列化により，さらなる性能効率の改善，開発の効率化を行う．この方法のメリットは，並列動作が可能な命令の発行ウィンドウを不連続に広げられること．

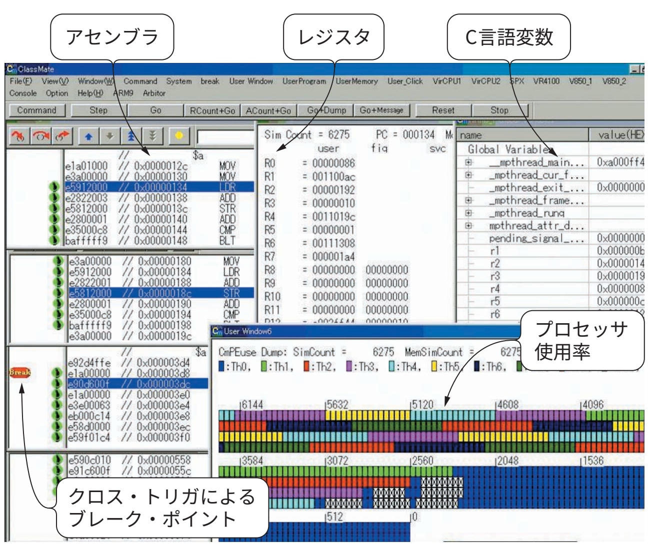


図7　マルチプロセッサのデバッグ環境
プロセッサの使用効率のモニタリングやプロセッサ間のクロス・トリガ機能によって，性能の最適化や効率の良いデバッグを行う必要がある．一例として，筆者らがマルチプロセッサの開発に使用しているツールのイメージを示す．

　このようなケースを防ぐため，プロセッサによっては個々のプロセッサの動作状態をほかのプロセッサに通知するしくみを持ったものがあります．このようなしくみを用いたデバッグ手法やツールの確立が必要となります．

　上述した課題を解決するには，マルチプロセッサを使っていくうえで必要となるソフトウェアや開発環境の基盤を整備しなければなりません．ひとたび基盤が整えば，組み込み機器の分野でもマルチプロセッサを使いこなしていくことは難しいことではありません．

　マルチプロセッサを使うことによって，組み込み機器ではこれまで考えられなかったような高性能，高機能機器が実現されていくものと考えています．

よしだ・まさやす
NECエレクトロニクス(株)

<筆者プロフィール>
吉田正康．1987年にNECに入社．以来AV機器向け，携帯端末向けのLSIを中心としたシステム・デザインを手がける．とくに，OSやミドルウェアなどのデバイスとソフトウェアのグレーゾーンを担当．現在は，カー・ナビゲーション・システムへのマルチプロセッサの展開を計画中．